Implementation and Evaluation of an SCA-Resistant Embedded Processor

Stefan Tillich, Mario Kirschbaum, Alexander Szekely

Implementation and Evaluation of an SCA-Resistant Embedded Processor

Stefan Tillich[1], Mario Kirschbaum[2], and Alexander Szekely[2]

[1] University of Bristol, Computer Science Department, Merchant Venturers Building, Woodland Road, BS8 1UB, Bristol, UK
tillich@cs.bris.ac.uk
[2] Graz University of Technology,
Institute for Applied Information Processing and Communications,
Inffeldgasse 16a, A–8010 Graz, Austria
{Mario.Kirschbaum,Alexander.Szekely}@iaik.tugraz.at

Abstract. Side-channel analysis (SCA) attacks are a threat for many embedded applications which have a need for security. With embedded processors being at the very heart of such applications, it is desirable to address SCA attacks with countermeasures which "naturally" fit deployment in those processors. This paper describes our work in implementing one such protection concept in an ASIC prototype and our results from a practical evaluation of its security. We are able to demonstrate that the basic principle of limiting the "leaking" portion of the processor works rather well to reduce the side-channel leakage. From this result we can draw valuable conclusions for future embedded processor design. In order to minimize the remaining leakage, the security concept calls for the application of a secure logic style. We used two concrete secure logic styles (iMDPL and DWDDL) in order to demonstrate this increase in security. Unfortunately, neither of these logic styles seems to do a particularly good job as we were still able to attribute SCA leakage to the secure-logic part of the processor. If a better suited logic style can be employed we believe that the overall leakage of the processor can be further reduced. Thus we deem the evaluated security concept as a viable method for protecting embedded processors.
Keywords: Side-channel analysis, SCA countermeasures, embedded processors, iMDPL, DWDDL.

1 Introduction

Two well-known methods for protecting cryptographic workloads on embedded processors against side-channel analysis (SCA) attacks are software countermeasures and cryptographic coprocessors with custom hardware protection. On the one hand, software countermeasures are the most flexible solution but might not address all attacks or bear a significant overhead in terms of running time or resources. On the other hand, coprocessors often only offer a limited range of functionality and inhibit even slight changes of parameters (e.g. new modes of

operation). An alternative approach is the incorporation of SCA countermeasures into the processor itself in order to find a trade-off between the flexibility of software countermeasures and the protection of hardware countermeasures.

There have been several proposals for adapting processors directly in order to increase their SCA resistance. Randomized execution of programs is the basic concept in non-deterministic processors as proposed by May *et al.* [11]. Regazzoni *et al.* [16] built an automated design flow for integrating custom functional units (FUs) implemented in a secure logic style into the processor. The MUTE-AES architecture proposed by Ambrose *et al.* [1] uses a second processor to balance out the AES operations of the first with the calculation of the same operations on inverted data. The balancing of the second processor is activated and terminated automatically by detecting certain sequences of instructions and it can be used for other tasks when no cryptographic operations are performed on the first processor. Recently, additional works regarding SCA protection of processors have been published. Nakatsu *et al.* [12] have investigated a processor modification in combination with software countermeasures for eliminating SCA vulnerabilities due to branches, addresses and intermediate values dependent on secret information. They propose to implement the ALU in a specific secure logic style (Random Switching Logic) and use masking of operands and results. Although the authors emphasize that their proposal can be applied to various operations and processor architectures, several issues regarding mask handling remain open. Barthe *et al.* [2] proposed to duplicate the processor data path by adding an additional register file and pipeline registers. Data in the original data path is masked while the second data path carries the corresponding masks. Before entering the FUs, data is unmasked and the according result is freshly masked. Data in memory is protected with a "static" mask.

Protecting a processor from SCA with a combination of masking and the application of secure logic styles has first been proposed by Tillich *et al.* in [19]. The original proposal was targeted at securing a specific set of cryptographic instructions. Based on the original idea, a full security concept covering a wide range of instructions and addressing issues of hardware limitations and secure task switching was then presented in [20]. As this work was conducted in context of the so-called Power-Trust project, we will refer to this security concept as the *Power-Trust security concept* in the remainder of this paper. The work of [20] also produced an FPGA prototype to validate the functionality and a preliminary SCA evaluation to demonstrate the principal protection offered by the concept (although the use of FPGA-style secure logic had been omitted).

In the present paper we present an ASIC implementation of the Power-Trust security concept based on the SPARC V8 Leon3 embedded processor. Our prototype includes several optional features proposed in [20]. For the critical portion of the processor (the so-called *secure zone*) we have used two different secure logic styles. Additionally, we have included a CMOS variant of the secure zone for reference. The goal of our work was threefold:

– Demonstrate the functionality and practicality of the concept.

- Test the utility of the optional features and develop possible improvements both regarding usage in software and simplification in hardware.
- Evaluate the SCA resistance of the prototype in order to estimate the potential protection offered by the Power-Trust security concept.

Although we have observed huge improvements in DPA attack resistance in our practical evaluation, the use of our chosen secure logic styles has turned out to be rather ineffective (leading only to little improvements in resistance at relatively high cost). Nevertheless, we deem our results as valuable insights into the issues of practical deployment of secure logic styles.

The rest of this paper is organized as follows. The Power-Trust security concept is described in Section 2. Our ASIC prototype implementation including a description of features of the secure zone as well as details of the implementation of the logic styles on ASIC is presented in Section 3. Section 4 discusses possible improvements of the whole security concept. A detailed security evaluation and the presentation of our SCA attack results is given in Section 5, followed by analyses of the results in Section 6. Conclusions are drawn in Section 7.

2 Description of the Power-Trust Security Concept

SCA attacks exploit the effect specific key-dependent data has on various observable physical characteristics of a device, e.g. timing or power consumption. We will denote such data as *critical data* in the following. The basic assumption for the Power-Trust security concept is that each handling of critical data is a potential target for an attacker. The general term handling hereby refers to both operations on the critical data as well as the movement of it through various data flow control elements (e.g. multiplexers) and storage units. For example, in a typical modern processor critical data are not only operated upon in a number of functional units but also have to pass through various pipeline stages, feedback paths, and storage elements like pipeline registers, register files, caches and memories. All these activities are reflected in the processor's power consumption and can be ultimately exploited in a power analysis attack.

The first step for counteracting SCA attacks is to minimize the effect of the critical data on the physical characteristics. Leakage in the timing of the processor can be addressed by eliminating key-dependent branches and avoiding other potentially variable-time operations like table lookups. The use of special cryptographic instructions (instruction set extensions) is very helpful in achieving this goal without sacrificing performance. Also, elimination of table lookups is expected to go a long way in reducing leakage in the power side channel, as memory operations are usually a significant contributor to the overall power consumption.

With the help of cryptographic instruction set extensions (ISEs) it is possible to confine actual operations on critical data exclusively to the FUs of the processor, which are usually the main part of its execute stage. Any other part of the processor then just moves around critical data without transforming it.

Based on this observation, the Power-Trust security concept applies a mask to all critical data which circulates outside of the FUs. The operands entering a FU have to be unmasked and the results leaving the FU have to be masked again. Therefore, arbitrary FUs can be supported easily. This is unlike most of the previous masking solutions, which require the development of special FUs to implement non-linear operations which transform masked values and mask.

Thus, direct leakage only emanates from the FUs which are required for implementing the cryptographic algorithm. Leakage from components which handle the masks (mask storage and mask generator) could also lead to higher-order weaknesses when combined with leakage from the masked values. In order to mitigate these vulnerabilities, all these components are implemented in a secure logic style. This portion of the processor is denoted as *secure zone*. From its functionality and interface, the secure zone is very similar to a conventional FU, which facilitates its implementation in secure logic and its integration into the processor. Conceptually, an attacker must overcome the protection of the secure logic style[3] in order to break the security of the processor.

The secure zone can only store a fixed number of masks, which determines the maximum size of the intermediate state of a cryptographic algorithm at any point in time. However, by using pseudo-randomly generated masks it is possible to store masks intermittently outside of the secure zone in a secure redundant representation. This mechanism allows to virtually extend the number of available masks and also to share the secure zone between several processes in a secure fashion. Other issues addressed in [20] are the way that masks and masked values are associated in the processor and the options for dealing with exceptional cases during runtime, e.g. when the maximum number of possible masks is reached. Those issues afford various solutions and we describe our concrete design choices in Section 3.

3 ASIC Prototype Implementation

Our prototype is based on the SPARC V8 Leon3 processor [4], which is a popular platform for academic research due to its high quality and tool support as well as its openness. It is a 32-bit embedded architecture with a large number of configuration options. Our most important enhancements are described in the following.

3.1 Features of the Secure Zone

When masked operands enter the secure zone, the processor needs a mechanism to identify the corresponding mask. Various options are possible but in our implementation we chose to introduce a custom addressing mechanism which is under explicit software control. Our reason for this choice was the huge degree of

[3] Either by attacking the unmasked critical data in the secure zone or with a higher-order attack on the masked values and their masks in the secure zone.

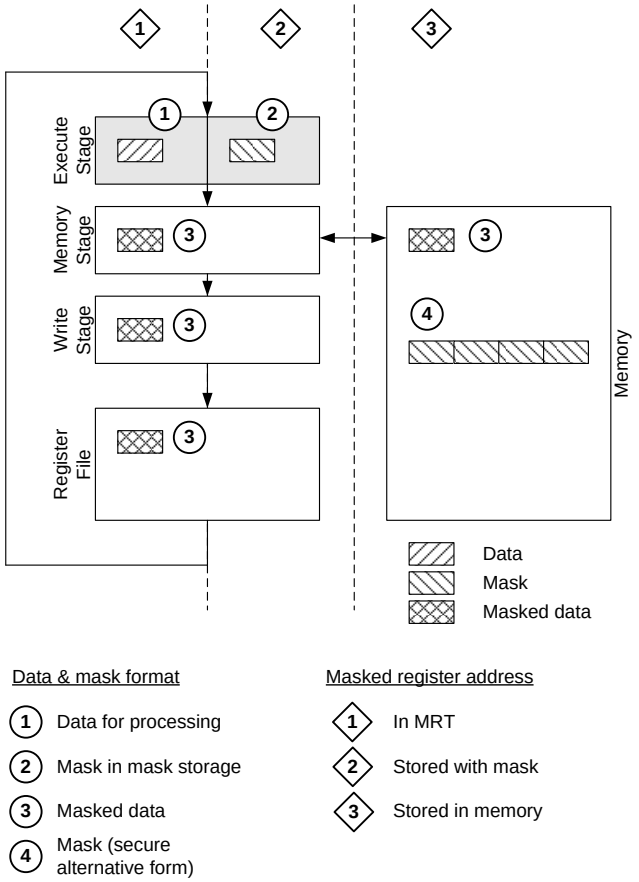


Fig. 1: Data representation and addressing in enhanced processor.

flexibility offered by this solution. Each mask and each masked value is associated with a custom 10-bit address, so that the mask and masked value with the same address belong together. As this address acts much like a register address, it is denoted as *masked register address*. Depending on the storage location of mask or masked value, the adjoint masked register address is handled differently.

Figure 1 depicts the different possible data representations of masked data and masks in various storage locations as well as the mechanisms used to keep the masked register address alongside them. Unmasked data only occurs in the secure zone's FUs (circled 1) and masks reside exclusively in the secure zone's mask storage unit (circled 2), where both are protected by the secure logic style (indicated by the gray shading). Masked data can circulate through the other pipeline stages, the register file, and memory (circled 3). Additionally, masks can be held in their secure redundant form in memory (circled 4). Masks and masked data must always be attached to their masked register address, so that the secure

zone can associate masked data with its corresponding mask. Masked data in the pipeline stages and register file (diamond 1) are principally attached to a logical register address. A custom hardware table called *Masked Register Table (MRT)* then maps this logical register address to a masked register address. The masks in the mask storage (diamond 2) are held automatically alongside their masked register address. On the other hand, masked data and masks in memory (diamond 3) must have their masked register address stored explicitly with them by the software.

The advantage of this masked register addressing scheme is that it can be used to emulate several different forms of software usage of the secure zone. For example, the software can choose to never write masks or masked values out to memory. In this case, the MRT can be initialized with a fixed mapping and the logical register address of a masked value effectively determines the corresponding mask.

The instruction result coming from the secure zone must be protected by a fresh 32-bit mask. When the processor handles a sequence of such protected instructions, a fresh mask is needed in every clock cycle. Furthermore, masks must be uniformly distributed and unpredictable by an attacker. For dealing with more masks than the mask storage of the secure zone can handle internally, it is also necessary to have a secure representation of masks which allows their subsequent reconstruction.

In order to balance all these (partly conflicting) requirements with an efficient design we have implemented our mask generator as a 127-bit maximum-length LFSR. We used the pentanomial $x^{127} + x^{87} + x^{59} + x^{37} + 1$ as the reduction polynomial because it facilitates parallelization in both forward and reverse direction. In normal operation, the LFSR advances 32 steps per cycle in order to produce a fresh 32-bit mask.

The mask generator also keeps track of the number of steps (denoted as *mask index*) it has taken from a specific initial state. In conjunction with a particular LFSR state, the mask index is a secure alternative representation of a mask. Software can read and write the LFSR state and can run the LFSR in forward or reverse direction towards a specific mask index in order to restore a mask. In a finished device, the LFSR should be seeded from a random number generator (RNG) to make the produced masks unpredictable and to prevent reset attacks. For our prototype, we did not integrate an RNG.

The secure zone can hold up to eight masks in its mask storage component. Metadata about stored mask like masked register address and mask index can be read by software. Exceptional conditions in the secure zone cause traps. Such exceptional conditions are a full mask storage, encounter of a masked operand whose mask is missing from the mask storage, and an overflow of the mask index counter in the mask generator. It is up to the software to either avoid those exceptional conditions altogether or to address them at runtime with specific trap handlers. Protected instructions of the secure zone encompass specific support for AES and ECC over GF(2^m) as well as a range of logic operations for bit-sliced implementations.

3.2 Implementation Details

We have used the UMC 0.18 µm standard-cell library FSA0A_C from Faraday [3] for implementation. Our original plan was to employ memory macros from this library to implement memory components like the caches. However, despite our best efforts we were unable to integrate those memory macros into our design flow within a reasonable time frame. Therefore, we used the synthesizer to infer those memory structures with flip-flops. As such inferred structures are significantly larger compared to macros, we had to reduce the size of instruction and data cache to the minimum of 1 KB each.

We have integrated secure zone blocks in three different logic styles in our prototype. We chose logic styles which are based on standard cell libraries and thus can be relatively easily implemented in a more or less automatic way but which still promised to deliver an increase in protection over CMOS. To this end we chose iMDPL [14] and DWDDL [23]. The third secure zone was implemented in CMOS to serve as a point of reference. Software can select a specific secure zone by writing to a custom configuration register. The inputs (including the clock signal) of the other two secure zones are held at zero in order to minimize their impact on the power profile.

The improved masked dual-rail precharge logic (iMDPL) style combines the dual-rail precharge (DRP) technique, a masking technique, and it takes precautions to prevent the effect of early propagation ([18]). The DRP technique avoids the occurrence of glitches ([15]) which may significantly reduce the side-channel resistance of implementations ([9] and [10]). This is achieved by introducing an evaluation and a precharge phase in each clock cycle. In the precharge phase, both wire rails of each signal are precharged. In the evaluation phase the wires switch to their corresponding values. The masking technique bypasses the need for special routing techniques of the circuitry. In [14] it has been shown that early propagation may cause a significant side-channel leakage in a masked DRP logic style. Thus, each iMDPL gate contains an evaluation-precharge detection unit (EPDU) which tries to prevent the premature evaluation/precharge of the gates.

A basic iMDPL AND gate is built with two three-input majority (MAJ) gates. MAJ gates are part of standard cell libraries, so there is no need to design and verify special security cells in the library. The iMDPL style can thus be implemented by a more or less straight forward search and replace processing step after design synthesis. Each iMDPL gate processes the masked values a_m, b_m ($\bar{a}_m, \bar{b}_m$) and the mask value m ($\bar{m}$). IMDPL is based on boolean masking, *i.e.* $a_m = a \oplus m$, and the mask value m is derived from a 64-bit maximum-length LFSR counter [13] in each clock cycle.

The area of a pure iMDPL implementation is usually increased by a factor of approximately 20. In our case, the area of the iMDPL secure zone is 406 kGE, compared to 19 kGE of the secure zone implemented in unprotected CMOS logic, which corresponds to a factor of 21. The whole Leon3 processor without any

secure zones has 582 kGE[4]. This results in a total area overhead of approximately 64% compared to a pure CMOS implementation of the SPARCV8 embedded processor.

We implemented iMDPL with knowledge that the security of the logic style might suffer from unbalanced wires ([17], [21]). Recent research has shown that the mask value in an iMDPL circuit can most probably be discovered and that also other wires within the circuitry might be affected by imbalances. Nevertheless, investigations of an ASIC implemented in iMDPL have shown that the logic style is able to increase the SCA resistance ([5]).

Double Wave Dynamic Differential Logic (DWDDL) [23] is an enhancement of WDDL [22] with the goal of solving its inherent balancing issues. Whereas WDDL requires techniques of balanced routing of its differential signals, DWDDL duplicates the layout of a normally routed WDDL circuit and inverts both the block's input as well as the inputs and outputs of all logic cells[5]. Thus, for each differential signal, a bit flip at one of the wires in the first WDDL block is counterbalanced by a bit flip on the other wire of this differential signal in the second WDDL block.

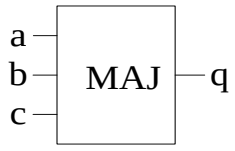
The area of the DWDDL secure zone is 260 kGE, which represents a factor of 14 compared to the secure zone implemented in unprotected CMOS logic. With respect to the whole Leon3 processor, the total area overhead is approximately 40%, which represents a relatively moderate increase in area in return for a significantly increased SCA resistance.

DWDDL has originally been proposed exclusively for use in FPGAs, as they allow to invert the functionality of logic cells simply by changing the contents of their lookup tables (LUTs). In case of ASIC implementations a swapping of cell functionality (*i.e.* substituting an AND cell with an OR cell) would result in significant changes in the circuitry due to differences in the structure of the cells. This would implicate differences between the two WDDL circuits and would hence introduce imbalances in the power consumption of the two circuits. We have developed a simple way to overcome this issues and to adopt DWDDL for ASIC implementations. Our approach is based on the observation that three-input MAJ gates can be seen as configurable two-input AND/OR gates (inputs a and b), where the value of the third MAJ input c determines the functionality, see Figure 2.

We started with the design synthesis utilizing conventional AND and OR gates. Afterwards we implemented WDDL based on MAJ gates by means of a search and replace processing step in which the standard AND and OR cells are replaced by corresponding WDDL cells (based on MAJ cells) and the whole netlist is transformed to a dual-rail circuitry. The third input of each MAJ gate

[4] The area of 582 kGE of the Leon3 processor includes everything except the three secure zones. Note the fact that we implemented the chip using inferred memory structures instead of smaller memory macros. Subtracting the inferred memories (488 kGE) and some conventional AES ISEs (6 kGE), the size of the bare processor is about 88 kGE.

[5] In practice, this means that AND gates become OR gates and vice versa.



MAJ as AND				MAJ as OR			
a	**b**	**c**	**q**	**a**	**b**	**c**	**q**
0	0	0	0	0	0	1	0
0	1	0	0	0	1	1	1
1	0	0	0	1	0	1	1
1	1	0	1	1	1	1	1

Fig. 2: Using a MAJ gate as configurable AND/OR gate. The gate functionality can be controlled by input c of the MAJ gate.

is hooked up to a global configuration signal. This WDDL netlist is then placed and routed. The resulting layout is duplicated and the second instance of the WDDL block is hooked up to the inverted inputs and the inverted configuration signal. This way we achieved two identically placed and routed WDDL circuits with a globally configurable cell functionality. We are aware that such a separation of the two WDDL circuits on the chip opens the door for localized EM measurements and thus represents a potential vulnerability.

A critical point for the SCA resistance of the DWDDL implementation is the timing of the two WDDL circuits. The propagation of the evaluation and precharge waves through the circuits has to start as exact as possible in both circuits, deviations would result in early propagation which would significantly affect the SCA resistance. Thus, we implemented the transformation cells CMOStoWDDL in a way that the clock signal acts as a trigger for the precharge/evaluation phases. Internally, the WDDL circuits are operated with a doubled clock speed due to the master-slave flip-flop structure implemented [22]. So we have two clock domains in the whole design: (1) the standard clock signal for all CMOS circuits and the iMDPL secure zone, which also serves as a precharge/evaluation signal and (2) a double-speed clock signal for the two WDDL circuits. We also had to take precautions when implementing the transformation cells for leaving the dual-rail circuit. Similar to inputs in the CMOStoWDDL cells, in case of WDDLtoCMOS the sensitive outputs of both WDDL circuits have to be synchronized to prevent a data leakage. Thus, we added a register stage to the output transformation which is synchronized by the clock signal.

Known weakness in DWDDL: Before the implementation of DWDDL we performed a theoretical investigation. Unfortunately, at this point we made a mistake and evaluated an incorrectly designed DWDDL cell: we correctly inverted the input signals of the complementary WDDL cell but we missed to exchange the logic gates AND and OR of the complementary WDDL cell. Our mistake disguised the occurrence of early propagation in DWDDL and resulted in an incorrect behavior of the complementary WDDL circuit. We recognized the mistake only shortly before the tape-out date, where we still had the time to correctly exchange the logic gates but we did not have enough time to switch to another logic style or to implement and improved version of DWDDL.

4 Possible Improvements

Experience with software development using the various features of the prototype has led to a number of possible improvements. Most of the features are intended to support manual software development for the enhanced processor. However, we expect that software generation should ultimately be pushed off to the toolchain in order to minimize the risk of programming errors. In such a case, many of the extra features of the secure zone could be simplified or eliminated in order to reduce its size and cost.

The biggest issue is the use of masked register addresses to associate masked data and masks. This leads to a high degree of flexibility for software which allows to move masked data freely between memory and any register. However, as allocation of masked data is expected to be ultimately handled by the toolchain, this flexibility does not seem to be required. Masked register addresses require elaborate address lookup logic and the inclusion of the MRT. By using the already present logical register address as mechanism to associate masked data and masks, the address lookup logic can be greatly simplified and the MRT can be eliminated altogether. This implies that masked data must be loaded back into the same logical register from which it has originally been stored to memory. However, we do not expect this to be a problem for the software.

The hardware traps are mainly intended for a very flexible but also rather slow mode of execution. This mode is intended for software developers who have little knowledge about the characteristics of the secure zone but still want to write code manually. By removing this mode of execution, the hardware trap mechanism can be simplified or even eliminated altogether.

Mask indices are maintained by the mask generator (*i.e.* its current "step" count) and the mask storage (*i.e.* the "step" count of each mask) with the sole purpose of having a secure redundant representation of masks. If the mask storage of the secure zone is deemed large enough to support all required cryptographic algorithms, these mask indexing mechanisms could be removed in order to reduce hardware overhead.

A more minor issue are the management instructions to read out metadata from the mask storage. In our prototype, it is not possible to get the metadata corresponding to a particular mask directly, which makes it necessary to read out all metadata and look for the desired entry. This problem could be easily remedied by a slight modification of the functionality of the management instructions.

5 Security Evaluation

We have compared various implementations of a single round of AES by means of classical correlation-based DPA attacks using Hamming weight (HW) and Hamming distance (HD) as power models. A standard software AES using only native SPARC V8 instructions and incorporating an S-box lookup table serves as our baseline version (SW-AES). Then we have the use of basic AES ISEs, which

eliminate the need for table lookups (AESREF). Furthermore, we implemented AES on each of the three secure zones (SZ_CMOS, SZ_IMPDL, SZ_DWDDL). We defined a DPA attack as successful if at least 9 of the 16 key bytes could have been revealed. We assumed that the remaining 7 key bytes can be revealed by a brute force attack within reasonable time. Our approach is based on common brute-force attacks on the 7-byte key of DES ([6] and [7]). Hence, for our estimations of the number of required power traces we used the ninth-highest correlation value of all correctly revealed key bytes.

In case of SW-AES we performed 10 000 measurements. According to our attack success metrics and according to the rule-of-thumb formula from [8] (Chapter 6.4.1) the number of required power traces to distinguish the correct key hypothesis from the incorrect ones is 263 (key byte 9 has the ninth-highest correlation value of 0.315). The SW-AES implemented in unprotected CMOS is highly vulnerable as expected. Due to the involvement of the MixColumns operation in the SW implementation, attacks using the HD power model would have resulted in rather expensive attacks based on 4 key bytes at once, and hence, these attacks have been omitted. Our results show that the AESREF implementation offers a very slightly increased protection compared to SW-AES. The number of required traces of the ninth-highest correlation value (0.0319) in case of HW is 27 200. The attacks using a HD power model resulted in significantly higher correlation values around 0.2, which corresponds to 700 required power traces.

We received very interesting results from the SZ_CMOS: it turned out that the secure zone implementation already has a significant effect on the SCA resistance, even without implementing the secure zone itself in a special logic style. According to our attack success metrics the number of required power traces in case of SZ_CMOS is around 130 000. The results of the attacks on the SZ_IMDPL show a further increase in SCA resistance: around 260 000 power traces are required to distinguish the ninth-highest correlation value. In case of SZ_DWDDL it turned out that the lower 16 bits of the four 32-bit words processed can not be clearly distinguished in a HW attack. As only eight key bytes could be revealed, we used the eighth-highest correlation value of 0.0064 in this case, which corresponds to 675 000 required power traces. By means of an HD attack on SZ_DWDDL, all 16 key bytes could have been revealed: the highest correlation value corresponds to a number of required power traces of approximately 5 200 000.

The results lead to the following conclusions: it shows that the confinement of leaking operations to a small part of the implemented processor significantly enhances the SCA resistance, even though the confined part is not implemented in a secure logic style. The results also show that the implemented logic styles still have an information leakage, therefore we can not conclusively show that the overall security will increase further with a reliably secure logic style. Nevertheless, the assumption that the elimination of the leakage in the confined part of the processor would also have a positive effect on the overall security still seems plausible.

6 Analysis of the Results

The results of the DPA attacks using Hamming weight power model are summarized in Table 1, the results using Hamming distance power model are summarized in Table 2.

The AES ISEs in the FU of the AESREF implementation are identical to the ISEs in the FU of the SZ_CMOS, *i.e.* the only difference between these two implementations is the masking of the result values leaving the SZ_CMOS. Hence, we assume that the significant HD leakage of AESREF is related to storing the unmasked results in the register file of the processor. In other words, the HD leakage of the SZ_CMOS is prevented due to the masking of the output operands.

The results of the SZ_IMDPL support the assumptions that routing imbalances within the circuitry limit the effectiveness of the logic style to a certain degree. Imbalances between complementary wires in a DRP circuit obviously cause signal-dependent differences in terms of power consumption as well as in terms of signal timings. In a conventional digital design which contains considerably large combinational structures such imbalances may influence the overall power consumption to a certain degree, which results in a data leakage.

Unfortunately, at this stage there are several questions unanswered which are marked for future work. During our evaluations it turned out that we are not able to fully resolve the open questions solely by means of measurements. It further turned out that we cannot perform logic simulations of the whole Leon3 processor within a reasonable time. Hence, we plan to perform simpler simulations of individual submodules in order to be able to investigate the processes within the Leon3 processor and the secure zones in more detail.

The proposal of Barthe *et al.* [2] is similar to the Power-Trust security concept: it tries to confine the leaking operations to a specific portion of the processor. However, protection of memory appears relatively weak as a constant mask added to all words is in principle not expected to deliver good protection against DPA attacks. The authors have performed practical evaluation of an FPGA implementation of an enhanced MicroBlaze clone. They used EM measurements at a number of points of the device. However, analysis of the acquired traces seems to have been done with classical difference-of-means DPA, which makes it hard to accurately estimate the minimal required number of traces for a successful attack from the resulting difference-of-means trace. Barthe *et al.* concluded that their proposal significantly reduces leakage from most parts of the pipeline but that leakage from the unprotected combinational logic in the FUs remains.

7 Conclusions

In this paper we presented an ASIC implementation of the Power-Trust security concept. Our prototype is based on the SPARC V8 Leon3 processor and has been produced in a 0.18 µm process technology from UMC. We demonstrated

the functionality and practicality of the security concept and we evaluated the SCA resistance of the ASIC prototype.

The results have shown that the use of basic AES ISEs, which eliminate the need for table lookups, already has a positive effect on the SCA resistance of an implementation. A further improvement of the SCA resistance has been achieved by the secure zone concept where critical operations are confined to a small part of the processor and all data outside of the secure zone is strictly masked. Even if the secure zone itself is not implemented in a secure logic style (as it is proposed by the Power-Trust security concept), the SCA resistance is significantly increased. A further increment in security is provided if the secure zone is implemented in a special logic style. Unfortunately, neither iMDPL nor DWDDL provide a very high level of SCA resistance, but our results indicate that the efficiency of the Power-Trust security concept could be further increased when implementing the secure zone in a well-performing logic style. Furthermore, restricting the implementation of a special logic style only to a small part of a processor tremendously reduces the overhead in terms of area and power which is associated with most of such logic styles.

Acknowledgements. The research described in this paper has been supported by the Austrian Science Fund (FWF) under grant number P22241-N23 ("Investigation of Implementation Attacks"), by EPSRC grant EP/H001689/1, and, in part by the European Commission through the ICT Programme under contract ICT-2007-216676 ECRYPT II. The information in this document reflects only the author's views, is provided as is, and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

References

1. J. A. Ambrose, S. Parameswaran, and A. Ignjatovic. MUTE-AES: A Multiprocessor Architecture to prevent Power Analysis based Side Channel Attack of the AES Algorithm. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2008*, pages 678–684. IEEE, 2008.
2. L. Barthe, P. Benoit, and L. Torres. Investigation of a Masking Countermeasure against Side-Channel Attacks for RISC-based Processor Architectures. In *2010 International Conference on Field Programmable Logic and Applications*, pages 139–144. IEEE Computer Society, 2010.
3. Faraday Technology Corporation. Faraday FSA0A_C 0.18 μm ASIC Standard Cell Library, 2004. Details available online at `http://www.faraday-tech.com`.
4. Gaisler Research. GRLIB IP Library User's Manual. Available online at `http://www.gaisler.com/products/grlib/grlib.pdf`, October 2010. Version 1.1.0 B4100.
5. M. Kirschbaum and T. Popp. Evaluation of a DPA-Resistant Prototype Chip. In *25th Annual Computer Security Applications Conference (ACSAC 2009), 7-11 December 2009, Honolulu, Hawaii, USA*, 2009.

6. S. S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, A. Rupp, and M. Schimmler. How to Break DES for 8,980. Workshop on Special-purpose Hardware for Attacking Cryptographic Systems - SHARCS 2006, April 3-4, Cologne, Germany, 2006.
7. S. S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler. Breaking Ciphers with COPACOBANA – A Cost-Optimized Parallel Code Breaker. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 101–118. Springer, 2006.
8. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks – Revealing the Secrets of Smart Cards*. Springer, 2007. ISBN 978-0-387-30857-9.
9. S. Mangard, T. Popp, and B. M. Gammel. Side-Channel Leakage of Masked CMOS Gates. In A. Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, February 2005.
10. S. Mangard, N. Pramstaller, and E. Oswald. Successfully Attacking Masked AES Hardware Implementations. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.
11. D. May, H. L. Muller, and N. P. Smart. Non-deterministic Processors. In V. Varadharajan and Y. Mu, editors, *Information Security and Privacy, 6th Australasian Conference, ACISP 2001, Sydney, Australia, July 11-13, 2001, Proceedings*, volume 2119 of *Lecture Notes in Computer Science*, pages 115–129. Springer, 2001.
12. D. Nakatsu, Y. Li, K. Sakiyama, and K. Ohta. Combination of SW Countermeasure and CPU Modification on FPGA against Power Analysis. In Y. Chung and M. Yung, editors, *Information Security Applications, 11th International Workshop, WISA 2010, Jeju Island, Korea, August 24-26, 2010, Revised Selected Papers*, volume 6513 of *Lecture Notes in Computer Science*, pages 258–272. Springer, 2011.
13. Peter Alfke. Xilinx Application note on Shift Registers and LFSR counters. Available online at `http://www.xilinx.com/support/documentation/application_notes/xapp052.pdf`, July 1996.
14. T. Popp, M. Kirschbaum, T. Zefferer, and S. Mangard. Evaluation of the Masked Logic Style MDPL on a Prototype Chip. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 81–94. Springer, September 2007. ISBN 978-3-540-74734-5.
15. T. Popp and S. Mangard. Masked Dual-Rail Pre-Charge Logic: DPA-Resistance without Routing Constraints. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 172–186. Springer, 2005.
16. F. Regazzoni, A. Cevrero, F.-X. Standaert, S. Badel, T. Kluter, P. Brisk, Y. Leblebici, and P. Ienne. A Design Flow and Evaluation Framework for DPA-Resistant Instruction Set Extensions. In C. Clavier and K. Gaj, editors, *Cryptographic Hardware and Embedded Systems – CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747

of *Lecture Notes in Computer Science*, pages 205–219. Springer, 2009. ISBN 978-3-642-04137-2.

17. P. Schaumont and K. Tiri. Masking and Dual-Rail Logic Dont Add Up. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 95–106. Springer, September 2007.
18. D. Suzuki and M. Saeki. Security Evaluation of DPA Countermeasures Using Dual-Rail Pre-charge Logic Style. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 255–269. Springer, 2006.
19. S. Tillich and J. Großschädl. Power-Analysis Resistant AES Implementation with Instruction Set Extensions. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 303–319. Springer, September 2007.
20. S. Tillich, M. Kirschbaum, and A. Szekely. SCA-Resistant Embedded Processors—The Next Generation. In *26th Annual Computer Security Applications Conference (ACSAC 2010), 6-10 December 2010, Austin, Texas, USA*, pages 211–220. ACM, 2010.
21. K. Tiri and P. Schaumont. Changing the Odds against Masked Logic. In E. Biham and A. M.Youssef, editors, *Selected Areas in Cryptography, 13th International Workshop, SAC 2006, Montreal, Quebec, Canada, August 17-18, 2006, Revised Selected Papers*, volume 4356 of *Lecture Notes in Computer Science*, pages 134–146. Springer, 2007. Available online at `http://rijndael.ece.vt.edu/schaum/papers/2006sac.pdf`.
22. K. Tiri and I. Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In *2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), 16-20 February 2004, Paris, France*, volume 1, pages 246–251. IEEE Computer Society, February 2004. ISBN 0-7695-2085-5.
23. P. Yu and P. Schaumont. Secure FPGA circuits using controlled placement and routing. In *Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis, Salzburg, Austria, September 30 - October 5, 2007*, pages 45–50. ACM Press, September 2007. ISBN 978-1-59593-824-4.

A Analysis Results

Target	SW-AES	AESREF	SZ_CMOS	SZ_IMPDL	SZ_DWDDL
Byte 1	0.227	0.0164*	0.0111	0.0066	0.0028*
Byte 2	0.298	0.162	<0.003*	0.0177	0.0025*
Byte 3	0.574	0.0244	0.0264	**0.0103**	0.0121
Byte 4	0.135	0.177	0.0177	0.0289	0.0243
Byte 5	0.230	0.0334	**0.0146**	0.0067	0.0024*
Byte 6	0.697	0.0162*	0.0074	0.0195	0.0018*
Byte 7	0.672	0.0878	0.0261	0.0080	**0.0064**
Byte 8	0.178	0.0294	0.0213	0.0289	0.0120
Byte 9	**0.315**	0.0222	0.0085	0.0059	0.0026*
Byte 10	0.564	0.0355	0.0067	0.0166	0.0018*
Byte 11	0.511	0.0240	0.0228	0.0078	0.0091
Byte 12	0.556	0.1296	0.0192	0.0268	0.0114
Byte 13	0.409	0.145	0.0050	0.0062	0.0028*
Byte 14	0.294	0.0201	0.0045*	0.0181	0.0019*
Byte 15	0.452	0.168	0.0237	0.0084	0.0155
Byte 16	0.113	**0.0319**	0.0240	0.0279	0.0098

Table 1: Correlation analysis results for the measurement series (Hamming weight model); correlation values marked with an asterisk indicate unsuccessful attacks; correlation values in bold represent the ninth-highest value (if at least nine key bytes could have been discovered).

Target	AESREF	SZ_CMOS	SZ_IMPDL	SZ_DWDDL
Byte 1 → 5	0.184	<0.003*	<0.002*	0.0019
Byte 3 → 7	0.211	<0.003*	<0.002*	0.0019
Byte 5 → 9	0.184	<0.003*	<0.002*	0.0020
Byte 6 → 10	0.200	<0.003*	<0.002*	0.0023
Byte 7 → 11	0.212	<0.003*	<0.002*	0.0021
Byte 8 → 12	0.221	<0.003*	<0.002*	0.0023
Byte 9 → 13	0.184	<0.003*	<0.002*	0.0017
Byte 10 → 14	0.201	<0.003*	<0.002*	0.0019
Byte 11 → 15	0.214	0.0314	<0.002*	0.0021
Byte 12 → 16	0.221	<0.003*	<0.002*	0.0021
Byte 14 → 2	0.201	<0.003*	<0.002*	0.0017
Byte 16 → 4	0.222	0.0239	<0.002*	0.0021

Table 2: Correlation analysis results for the measurement series (Hamming distance model); correlation values marked with an asterisk indicate unsuccessful attacks.